Building a Real-Time Data Server in Excel 2002

Paul Cornell
Microsoft Corporation

July 2001

Applies to:
Microsoft® Excel 2002

Summary: This article shows you how to build a real-time data server in Microsoft Excel 2002.

Download Odc_xlrtdbuild.exe.

Contents

Introduction

Microsoft Excel 2002 now provides you with a new way to view and update data in real time. This real-time data (RTD) feature is great for working with constantly-changing data such as stock quotes, currency exchange rates, inventory levels, price quotes, weather information, sports scores, and so on.

In the past, developers have had to rely on technologies such as Dynamic Data Exchange (DDE) to access real-time data sources. DDE has a different function format from standard Excel functions and wasn't designed for getting real-time data into Excel in a robust and high-performance way. The RTD feature overcomes these issues.

In this article, I explain how you can develop RTD solutions, and I provide you with a fully-functioning RTD example, complete with source code available as a download that accompanies this article.

Before we begin, you should be familiar with the notion of a *real-time data source*, an *RTD server*, and a *topic*.

- A *real-time data source* can be any source of data that can be accessed programmatically. Some examples of common real-time data sources are Microsoft Access and Microsoft SQL Server databases, and XML data files.
- An *RTD server* is a Component Object Model (COM) Automation server that implements the **IRtdServer** interface. Excel uses the RTD server to communicate with a real-time data source.
- A *topic* is a string (or a set of strings) that uniquely identifies a piece of data that resides in a real-time data source. A topic may exist by itself or it may reside in a hierarchy of *leaves*. For example, a topic could be as simple as **"MSFT"** or **"Address"**, but could also be more complex, such as **"MSFT", "BID_PRICE"** or **"Employee", "SSN", "Address"**. A good analogy is to think of a topic as a file name, with these files residing in folders. So the "BID_PRICE" file would reside in the "MSFT" folder, and the "Address" file would reside in the "SSN" subfolder, which in turn would reside in the "Employee" folder. The RTD server passes the topic to the real-time data source and receives the value of the topic from the real-time data source; the RTD server then passes the value of the topic to Excel for display. For example, the RTD server passes the topic "MSFT" to the real-time data source, and the RTD server receives the topic's value of "$72.12" from the real-time data source. The RTD server then passes the topic's value to Excel for display.

The Now & Today RTD Server

You can try out a simple RTD server called the Now & Today RTD server that is available in the sample download that accompanies this article. To try out the Now & Today RTD server:

1. Locate the file RTDTime.dll in the sample download.
2. On the **Start** menu, click **Run**. In the **Open** list, type **regsvr32.exe "C:\Path-to-RTD-Server\RTDTime.dll"**, where **Path-to-RTD-Server** is the path to the RTDTime.dll file on your local computer.

   Note On computers running Microsoft Windows® 98 or later, regsvr32.exe is located in the System folder. On computers running Microsoft Windows NT® 4.0 or later, regsvr32.exe is located in the System32 folder.

3. Start a new, blank Excel workbook and type the following function into cell A1: **=RTD ("RTDTime.RTD",,"Now")**
4. Right-click cell A1 and click **Format Cells**.
5. On the **Number** tab, in the **Category** list, click **Time**. In the **Type** list, click the entry that looks similar to **1:30:55 PM**, and then click **OK**. The time should automatically update every few seconds.

If you see the value #N/A in cell A1, make sure that you've successfully registered the RTDTime DLL on your computer, that you've typed the **RTD** function exactly as shown in step 3 above, and that your security level is not set to **High** (see the Security section below for more information).

Under the Hood

So how does the interaction occur between Excel, the RTD server, and the real-time data source?

On an Excel worksheet, a user types the **RTD** function into a cell, specifying the name of the RTD server, the computer on which the RTD server is running, and the topic name.

When the **RTD** function is called, Excel instantiates the RTD server (a COM Automation server that implements the **IRtdServer** interface) on the specified server by calling the RTD server's **ServerStart** method.

Next, the **ConnectData** method tells the real-time data source that Excel is "connected" to the specific topic, and Excel retrieves the topic's initial value.

When the real-time data source updates a topic, Excel uses an **IRTDUpdateEvent** callback object's **UpdateNotify** method to notify the RTD server that new data is available (the **IRTDUpdateEvent** callback object is instantiated by the RTD server as a liaison between itself and the real-time data source). After the throttle interval has expired (the default is two seconds, but can be changed through the **RTD** object's **ThrottleInterval** property in Excel), Excel calls the **RefreshData** method to get the new value for the specified topic.

When the topic is no longer needed (for example, the user removes the reference to the topic from the worksheet), Excel calls the **DisconnectData** method to remove the topic from its list of topics that it is monitoring.

When the RTD server is no longer needed (for example, the user quits Excel), Excel calls the **ServerTerminate** method, and then the RTD server terminates itself.

Security

One of the most common initial problems you encounter when you are using the **RTD** function in Excel is a problem with your security setting.

- If you have your security level in Excel set to **High** (the default setting), then no RTD servers will be available unless they are digitally signed and trusted. If the security level is set to **High**, then all you see is #N/A in cells that use the **RTD** function, and Excel provides no additional explanation as to why the **RTD** functions aren't working.
- If you set your security level in Excel to **Medium**, Excel asks you if you want to run any RTD servers that are not digitally signed and trusted and that are referenced by **RTD** functions.
- If you set your security level to **Low**, then all RTD servers run, regardless of whether they are digitally signed and trusted. Of course, setting your security level to **Low** presents a considerable security risk.

To view or change your security level in Excel, on the **Tools** menu, point to **Macro**, click **Security**, and click the **Security Level** tab.

Syntax

The syntax for the Excel **RTD** worksheet function is:

```
=RTD(ProgID, Server, String1, String2, ... String28)
```

- The *ProgID* parameter is a required **String** value representing the programmatic ID (ProgID) of the RTD server.
- The ***Server*** parameter is a required **String** value representing the name of the computer on your intranet (using Distributed COM (DCOM)) on which the RTD server is running. If the RTD server is running on the local computer, leave this parameter blank or use two quotation marks ("").

  Note When you use the **RTD** method of the **WorksheetFunction** object, you cannot leave the *Server* parameter blank; you must use two quotation marks to represent the local computer.

- The *String1* through *String28* parameters represent topics to be sent to the RTD server. Only the *String1* parameter is required; the *String2* through *String28* parameters are optional. There is a limit of 28 parameters, and in most cases, only the *String1* parameter is used. The actual values for the *String1* through *String28* parameters depend on the requirements of the real-time data server.

No RTD servers are shipped with Microsoft Office XP. If you haven't installed any real-time data servers or you use the wrong syntax for the **RTD** function, you get the error message #NAME? or #N/A in any cell that references the **RTD** function.

As stated in the Introduction, in order for the Excel **RTD** function to use the RTD server, the RTD server must implement the **IRtdServer** interface. The members of the **IRtdServer** interface are described in the following table:

IRtdServer interface method	**Description**
ConnectData(*TopicID*, *Strings*, *GetNewValues*)	This method is called whenever Excel requests new topics from the RTD server. The *TopicID* parameter is a required **Long** value that represents a unique, arbitrary value automatically assigned by Excel (for example, "34") that identifies the topic. The *Strings* parameter is a required array of one or more **Variant** values that the user enters into the **RTD** function to uniquely identify the topic. This **Variant** array should always be an array of **String** values. The *GetNewValues* parameter is a required Boolean value; **True** if new values are to be retrieved. Leave the *GetNewValues* parameter alone if you want Excel to use the previous value that it had saved with the Excel spreadsheet.
DisconnectData(*TopicID*)	This method is called whenever Excel no longer requires a specific topic. The *TopicID* parameter is a required **Long** value that represents the arbitrary, unique value automatically assigned by Excel in the **ConnectData** method that identifies the topic.
Heartbeat	If the RTD server is no longer able to process **RefreshData** method calls, the **Heartbeat** method enables Excel to pop up a dialog box that says "The real-time data server 'XYZ' is not responding. Would you like Microsoft Excel to attempt to restart the server?"

	This method returns a **Long** value; a value of 1 indicates that the real-time data server connection still exists, and a value of zero (0) or a negative value indicates that the real-time data server connection no longer exists.
RefreshData(*TopicCount*)	After a call from the **IRTDUpdateEvent** callback object's **UpdateNotify** method, this method is called by Excel to pull new values from the real-time data server (also known as a *topic refresh*). This method returns a two-dimensional array of **Variant** values. The first dimension represents a list of topic IDs; these topic IDs map to the ***TopicID*** parameter in the **ConnectData** method above. This is how Excel associates topics with data. The second dimension represents the values associated with the topic IDs. The *TopicCount* parameter is a required **Long** value that the RTD server provides; it represents the number of elements returned in the array of **Variant** values.
ServerStart(*CallbackObject*)	This method is called when Excel requests the first topic from the real-time data server. This method returns a **Long** value; a value of 1 indicates a successful request, and a value of zero (0) or a negative value indicates a failed request. This method call is immediately followed by a call to the **ConnectData** method. The ***CallbackObject*** parameter is a required **IRTDUpdateEvent** callback object that the RTD server uses to notify Excel when it should gather updates from the RTD server through the **IRTDUpdateEvent** callback object's **UpdateNotify** method.
ServerTerminate	This method is called when Excel no longer requires topics from the RTD server (for example, the user quits Excel).

The members of the **IRTDUpdateEvent** callback object are described in the following table:

IRTDUpdateEvent callback object member	**Description**
HeartbeatInterval property	This property returns or sets the time interval between RTD server updates. This property returns a **Long** value that represents the number of milliseconds between RTD server updates; this property cannot be set below the default of 15,000 milliseconds, because of the standard 15-second RTD server timeout.
Disconnect method	This method tells Excel that the real-time data source will be disconnecting from the RTD server. Excel can use this method to take some type of action before it loses its RTD connection.
UpdateNotify method	This method is called by the RTD server when it has a new value for one or more topics.

You can use development tools such as Microsoft Visual Basic® 6.0 Professional or Enterprise Edition to create an RTD server. You cannot create an RTD server by using the Visual Basic for Applications (VBA) development environment in Excel, nor can you create an RTD server by using the COM Add-In Designer that is included with

Microsoft Office XP Developer.

Let's build on what you've learned so far by creating an RTD server.

Solution: Wide World Importers Price List

This RTD solution is based on a fictitious company named Wide World Importers. Wide World Importers imports and sells fine furniture from around the world. Their merchandise sales prices change rapidly as foreign currencies change and foreign goods are imported into their Seattle warehouse. In this example, we will create an RTD server that updates the prices of Wide World Importers' price list every five seconds. For this example, pretend that this price list takes the form of an XML file that is generated from an XML-compliant database server (such as Microsoft SQL Server 2000). You could easily modify the solution to connect directly to a SQL Server or Microsoft Access database.

To create the RTD server:

1. Start Visual Basic 6.0 Professional or Enterprise Edition on your local computer where Excel 2002 and the MSXML 3.0 parser are installed (alternatively, you can skip down to step 14 to bypass entering all of the code).
2. In the **New Project** dialog box, click the **ActiveX DLL** icon, and then click **OK**.
3. In the **Properties** window, change the **(Name)** property of Class1 to **XMLRTD**.
4. On the **Project** menu, click **References**. Check the **Microsoft Excel 10.0 Object Library** box and the **Microsoft XML, v3.0** box, and then click **OK**.
5. In the **Code** window for the XMLRTD class, type the following:

```
Option Explicit

Implements Excel.IRtdServer

Private mcolTopics As Collection
Private Const SUCCESS = 1

Private Function IRtdServer_ConnectData(ByVal TopicID As Long, _
        Strings() As Variant, GetNewValues As Boolean) _
        As Variant

    On Error Resume Next

    Dim objTopic As New CTopic

    mcolTopics.Add Item:=objTopic, Key:=CStr(TopicID)
    objTopic.TopicID = TopicID
    objTopic.TopicString = Strings(0)

    gobjXMLDoc.Load xmlSource:=XML_FILE_PATH

    Set gobjXMLNode = gobjXMLDoc.selectSingleNode _
        (ROOT_NODE & LCase(Strings(0)))
    objTopic.TopicValue = gobjXMLNode.Text

    IRtdServer_ConnectData = objTopic.TopicValue

End Function

Private Sub IRtdServer_DisconnectData _
        (ByVal TopicID As Long)

    mcolTopics.Remove Index:=CStr(TopicID)

End Sub

Private Function IRtdServer_Heartbeat() As Long

    IRtdServer_Heartbeat = SUCCESS

End Function
```

```
Private Function IRtdServer_RefreshData _
        (TopicCount As Long) As Variant()

    Dim objTopic As CTopic
    Dim intArrayCounter As Integer
    ReDim avarUpdates(0 To 1, 0 To mcolTopics.Count - 1) As Variant

    For Each objTopic In mcolTopics
        objTopic.Update
        avarUpdates(0, intArrayCounter) = objTopic.TopicID
        avarUpdates(1, intArrayCounter) = objTopic.TopicValue
        intArrayCounter = intArrayCounter + 1
    Next objTopic

    TopicCount = mcolTopics.Count

    IRtdServer_RefreshData = avarUpdates

End Function

Private Function IRtdServer_ServerStart _
        (ByVal CallbackObject As Excel.IRTDUpdateEvent) _
        As Long

    Set gobjCallback = CallbackObject
    Set mcolTopics = New Collection
    Set gobjXMLDoc = New MSXML2.DOMDocument30

    glngTimerID = SetTimer(hWnd:=0, nIDEvent:=0, _
        uElapse:=TIMER_INTERVAL, lpTimerFunc:=AddressOf TimerCallback)

    If glngTimerID > 0 Then IRtdServer_ServerStart = SUCCESS

End Function

Private Sub IRtdServer_ServerTerminate()

    Dim objTopic As CTopic

    For Each objTopic In mcolTopics
        mcolTopics.Remove Index:=CStr(objTopic.TopicID)
    Next objTopic

    Set objTopic = Nothing

    Call KillTimer(hWnd:=0, nIDEvent:=glngTimerID)

End Sub
```

6. On the **Project** menu, click **Add Class Module** and click **OK**.
7. In the **Properties** window, change the **(Name)** property of Class1 to **CTopic**.
8. In the **Code** window for the CTopic class, type the following:

```
Option Explicit

Private mlngTopicID As Long
Private mstrTopicString As String
Private mvarValue As Variant

Friend Property Let TopicID(lngID As Long)

    mlngTopicID = lngID

End Property

Friend Property Get TopicID() As Long

    TopicID = mlngTopicID
```

```
End Property

Friend Property Let TopicString(strTopic As String)

    strTopic = LCase(strTopic)
    mstrTopicString = strTopic

End Property

Friend Sub Update()

    On Error Resume Next

    gobjXMLDoc.Load xmlSource:=XML_FILE_PATH

    Set gobjXMLNode = gobjXMLDoc.selectSingleNode(queryString:=ROOT_NODE & mstrTopicString)

    mvarValue = gobjXMLNode.Text

End Sub

Friend Property Get TopicValue() As Variant

    TopicValue = mvarValue

End Property

Friend Property Let TopicValue(strTopicValue As Variant)

    mvarValue = strTopicValue

End Property
```

9. On the **Project** menu, click **Add Module**.
10. In the **Properties** window, change the **(Name)** property of Module1 to **modGlobals**.
11. In the **Code** window for the modGlobals module, type the following:

```
Option Explicit

Public gobjXMLDoc As MSXML2.DOMDocument30
Public gobjXMLNode As MSXML2.IXMLDOMNode
Public gobjCallback As Excel.IRTDUpdateEvent
Public Const XML_FILE_PATH As String = "C:\Program Files\RTD Server\RTDXML.xml"
Public Const ROOT_NODE As String = "//prices/"
Public Const TIMER_INTERVAL = 5000
Public glngTimerID As Long

Public Declare Function SetTimer Lib "user32" (ByVal hWnd As Long, _
    ByVal nIDEvent As Long, ByVal uElapse As Long, _
    ByVal lpTimerFunc As Long) As Long

Public Declare Function KillTimer Lib "user32" (ByVal hWnd As Long, _
    ByVal nIDEvent As Long) As Long

Public Sub TimerCallback(ByVal hWnd As Long, ByVal uMsg As Long, _
    ByVal idEvent As Long, ByVal dwTime As Long)

    gobjCallback.UpdateNotify

End Sub
```

12. Save the project as **WideWorldRTD.vbp**.
13. On the **File** menu, click **Make WideWorldRTD.dll**.
14. On the **Start** menu, click **Run**. In the **Open** list, type **regsvr32.exe "C:\Path-to-RTD-Server\WideWorldRTD.dll"**, where **Path-to-RTD-Server** is the path to the WideWorldRTD.dll file on your local computer. (If you skipped to this step from step 1 above, locate the WideWorldRTD.dll file included with

the sample download that accompanies this article.)

To set up the real-time data source:

Open Notepad, type the following code into a text file (or use the file RTDXML.xml in the sample download that accompanies this article), and save the text file on your local computer in the path C:\Program Files\RTD Server\RTDXML.xml:

```
<?xml version="1.0"?>
<prices>
   <chair>$29.95</chair>
   <lamp>$49.95</lamp>
   <table>$99.95</table>
</prices>
```

To set up Excel 2002 to display real-time data:

Start a new, blank Excel workbook and enter the following values:

Cell	Value
A1	=RTD("WideWorldRTD.XMLRTD",,"chair")
A2	=RTD("WideWorldRTD.XMLRTD",,"lamp")
A3	=RTD("WideWorldRTD.XMLRTD",,"table")

To make updates to the real-time data:

1. Using Notepad, open the RTDXML.xml file located in the C:\Program Files\RTD Server\ folder.
2. Change the values of the <chair>, <lamp>, or <table> tags, and save the RTDXML.xml file.
3. Within five seconds, these values will automatically overwrite the existing values in the Excel spreadsheet.

> **Note** This action may take longer than 5 seconds depending on the value of the **RTD** object's **ThrottleInterval** property in Excel.

Exploring the Solution

Here's how the solution works:

1. When the user enters the first **RTD** function call `(=RTD("WideWorldRTD.XMLRTD",,"chair"))` into cell A1, Excel makes a call to the **ServerStart** method in the WideWorldRTD DLL. In the **ServerStart** method, the RTD server sets up the **IRTDUpdateEvent** callback object to receive update events from the XML data file, gets ready to connect to the XML data file, and initializes the five-second update timer.
2. Next, Excel uses the **ConnectData** method to add the "chair" topic to the collection of topics currently being monitored. Excel assigns the "chair" topic an arbitrary, unique value (for example, "7") that Excel uses as a identifier to get and update the topic's value (because there could be several instances of the "chair" topic in one or more cells in one or more workbooks on the user's computer). Then Excel gets the initial value of the <chair> tag in the XML file. Finally, Excel stores the topic ID of "7", the topic string of "chair", and the topic value of "$29.95" in the collection of topics currently being monitored.
3. Every five seconds, the Windows API function **SetTimer** calls the custom **TimerCallback** method. The **TimerCallback** method sends a message to the **IRTDUpdateEvent** callback object's **UpdateNotify** method to check the data in the XML file for updates. If there are updates, Excel eventually calls the **RefreshData** method to get new topic data values. (The **TimerCallback** method could be optimized more by only calling the **UpdateNotify** method when a change is made to the XML file, rather than calling the **UpdateNotify** method every five seconds.)
4. The **RefreshData** method prepares a two-dimensional array of topic IDs and topic values that correspond to these topic IDs that were assigned by Excel in the **ConnectData** method. The **RefreshData** method then queries the XML file again for updated data values and uses the two-dimensional array to write the new data values to the topic in the collection of topics being monitored. For instance, if the "chair" topic value changed to "$39.95", the **RefreshData** method would overwrite the value of "$29.95" that was originally captured in the **ConnectData** method (or a previous **RefreshData** method call).
5. If the topic is no longer needed (for example, the user deletes the **RTD** function call from cell A1), the

DisconnectData method removes the topic from the collection of topics being monitored.

6. If the connection to the real-time data source is no longer needed (for example, the user quits all running instances of Excel), the **ServerTerminate** method removes the entire collection of topics being monitored and stops the five-second update timer.

Summary

Microsoft Excel 2002 now provides you with a new way to view and update constantly changing data in real time. Using the Office development skills you already know, you can build solutions that let Excel interact with real-time data sources. In this article, you learned how to develop, deploy, and use RTD solutions in Excel.